Origin of anti-sickling activity via QSAR modelling

Chuleeporn Phanus-umporn, Saw Simeon, Watshara Shoombuatong and Chanin Nantasenamat

July 7, 2016

Reading the data file, removing the the column having Standard Deviation equal to 0 as well as intercorrelation higher than of 0.7

```
library(caret)
df <- read.csv("data.csv")
Activity <- df$Activity
descriptors <- df[, 2:ncol(df)]
data <- descriptors[, which(!apply(descriptors, 2, sd) == 0)]
raw <- cor(data)
raw_2 <- raw[1: ncol(raw), 1:ncol(raw)]
high <- findCorrelation(raw_2, cutoff = 0.7)
filtered_descriptors <- data[, -high]
real_data <- cbind(Activity, filtered_descriptors)
```

Fuction for building classification model using random forest 100 times (oversampling, stratified data splitting, stastical assessment including accuracy, sensitivity, specificity and matthew correlation coefficient)

```
RF_training_classification <- function(x) {
  library(parallel)
  library(doSNOW)
  cl <- makeCluster(8)
  registerDoSNOW(cl)

  ok <- list(100)
  ok <- foreach(i = 1:100 ) %dopar% {

    data <- x
    active <- subset(data, Activity == "active")
    inactive <- subset(data, Activity == "inactive")
    active <- dplyr::sample_n(active, size =  83, replace = TRUE)
    data <- rbind(active, inactive)
    trainIndex <- sample(nrow(data), size = as.integer(nrow(data) * 0.8),
                         replace = FALSE)
    train <- data[trainIndex, ]
    test <- data[-trainIndex, ]
    model_train <- ranger::ranger(Activity~., data = train, write.forest = TRUE, save.memory = TRUE)
    rm(ctrl)
    rm(rf)
    rm(data)
    rm(trainIndex)
    actual <- train$Activity
    prediction <- predict(model_train, train)
```

```
    prediction <- prediction$predictions
    rm(train)
    rm(test)
    rm(model_train)
    results <- caret::confusionMatrix(prediction, actual)
    results <- results$table
    results <- as.numeric(results)
    rm(prediction)
    rm(actual)
    ok[[i]] <- results



  }
  return(ok)
  stopCluster(cl)
}


mean_and_sd <- function(x) {
  c(round(mean(x, na.rm = TRUE), digits = 2),
    round(sd(x, na.rm = TRUE), digits = 2))
}

RF_train_classification <- function(x) {
  ok <- RF_training_classification(x)
  results <- data.frame(ok)
  rm(ok)
  data <- data.frame(results)
  rm(results)
  m = ncol(data)
  ACC  <- matrix(nrow = m, ncol = 1)
  SENS  <- matrix(nrow = m, ncol = 1)
  SPEC  <-matrix(nrow = m, ncol = 1)
  MCC <- matrix(nrow = m, ncol = 1)

  for(i in 1:m){
    ACC[i,1]  = (data[1,i]+data[4,i])/(data[1,i]+data[2,i]+data[3,i]+data[4,i])*100
    SENS[i,1]  =  (data[4,i])/(data[3,i]+data[4,i])*100
    SPEC[i,1]  = (data[1,i]/(data[1,i]+data[2,i]))*100
    MCC1      = (data[1,i]*data[4,i]) - (data[2,i]*data[3,i])
    MCC2      =  (data[4,i]+data[2,i])*(data[4,i]+data[3,i])
    MCC3      =  (data[1,i]+data[2,i])*(data[1,i]+data[3,i])
    MCC4  =  sqrt(MCC2)*sqrt(MCC3)


    MCC[i,1]  = MCC1/MCC4
  }
  results_ACC <- mean_and_sd(ACC)
  results_SENS <- mean_and_sd(SENS)
  results_SPEC <- mean_and_sd(SPEC)
  results_MCC <- mean_and_sd(MCC)
  rm(ACC)
```

```
  rm(SENS)
  rm(SPEC)
  rm(MCC)
  rm(data)
  rm(m)
  results_all <- (data.frame(c(results_ACC, results_SENS, results_SPEC, results_MCC)))
  rownames(results_all) <- c("ACC_Mean", "ACC_SD", "Sens_Mean", "Sens_SD", "Spec_Mean", "Spec_SD",
                             "MCC_Mean", "MCC_SD")
  rm(results_ACC)
  rm(results_SENS)
  rm(results_SPEC)
  rm(results_MCC)
  names(results_all) <- c("Training Performance")
  return(results_all)
}

RF_testing_classification <- function(x) {
  library(parallel)
  library(doSNOW)
  cl <- makeCluster(8)
  registerDoSNOW(cl)

  output <- list(100)
  output <- foreach(i = 1:100 ) %dopar% {

    data <- x
    active <- subset(data, Activity == "active")
    inactive <- subset(data, Activity == "inactive")
    active <- dplyr::sample_n(active, size =  83, replace = TRUE)
    data <- rbind(active, inactive)
    trainIndex <- sample(nrow(data), size = as.integer(nrow(data) * 0.8),
                         replace = FALSE)
    train <- data[trainIndex, ]
    test <- data[-trainIndex, ]
    model_train <- ranger::ranger(Activity~., data = train, write.forest = TRUE, save.memory = TRUE)
    rm(ctrl)
    rm(rf)
    rm(data)
    rm(trainIndex)
    actual <- test$Activity
    prediction <- predict(model_train, test)
    prediction <- prediction$predictions
    rm(train)
    rm(test)
    rm(model_train)
    results <- caret::confusionMatrix(prediction, actual)
    results <- results$table
    results <- as.numeric(results)
    rm(prediction)
    rm(actual)
    output[[i]] <- results
```

```r
  }
  return(output)
  stopCluster(cl)
}


mean_and_sd <- function(x) {
  c(round(mean(x, na.rm = TRUE), digits = 2),
    round(sd(x, na.rm = TRUE), digits = 2))
}

RF_test_classification <- function(x) {
  ok <- RF_testing_classification(x)
  results <- data.frame(ok)
  rm(ok)
  data <- data.frame(results)
  rm(results)
  m = ncol(data)
  ACC  <- matrix(nrow = m, ncol = 1)
  SENS  <- matrix(nrow = m, ncol = 1)
  SPEC  <-matrix(nrow = m, ncol = 1)
  MCC <- matrix(nrow = m, ncol = 1)

  for(i in 1:m){
    ACC[i,1]  = (data[1,i]+data[4,i])/(data[1,i]+data[2,i]+data[3,i]+data[4,i])*100
    SENS[i,1]  =  (data[4,i])/(data[3,i]+data[4,i])*100
    SPEC[i,1]  = (data[1,i]/(data[1,i]+data[2,i]))*100
    MCC1      = (data[1,i]*data[4,i]) - (data[2,i]*data[3,i])
    MCC2      =  (data[4,i]+data[2,i])*(data[4,i]+data[3,i])
    MCC3      =  (data[1,i]+data[2,i])*(data[1,i]+data[3,i])
    MCC4  =  sqrt(MCC2)*sqrt(MCC3)


    MCC[i,1]  = MCC1/MCC4
  }
  results_ACC <- mean_and_sd(ACC)
  results_SENS <- mean_and_sd(SENS)
  results_SPEC <- mean_and_sd(SPEC)
  results_MCC <- mean_and_sd(MCC)
  rm(ACC)
  rm(SENS)
  rm(SPEC)
  rm(MCC)
  rm(data)
  rm(m)
  results_all <- (data.frame(c(results_ACC, results_SENS, results_SPEC, results_MCC)))
  rownames(results_all) <- c("ACC_Mean", "ACC_SD", "Sens_Mean", "Sens_SD", "Spec_Mean", "Spec_SD",
                             "MCC_Mean", "MCC_SD")
  rm(results_ACC)
  rm(results_SENS)
  rm(results_SPEC)
  rm(results_MCC)
  names(results_all) <- c("Testing Performance")
```

```
  return(results_all)
}



RF_10_CV <- function(x){
  library(parallel)
  library(doSNOW)
  cl <- makeCluster(8)
  registerDoSNOW(cl)

  results <- list(100)
  results <- foreach(i = 1:100 ) %dopar% {
    data <- x
    active <- subset(data, Activity == "active")
    active <- dplyr::sample_n(active, size = 83, replace = TRUE)
    inactive <- subset(data, Activity == "inactive")
    data <- rbind(active, inactive)
    trainIndex <- sample(nrow(data), size = as.integer(nrow(data) * 0.8),
                         replace = FALSE)
    train <- data[trainIndex, ]
    test <- data[-trainIndex, ]
    k = 10
    index <- sample(1:k, nrow(train), replace = TRUE)
    folds <- 1:k
    myRes <- data.frame()
    for (j in 1:k) {
      training <- subset(train, index %in% folds[-j])
      testing <- subset(train, index %in% c(j))
      model_train <- ranger::ranger(Activity~., data = training, write.forest = TRUE, save.memory = TRU
      prediction <- predict(model_train, testing)
      prediction <- prediction$prediction
      actual <- testing$Activity
      ok <- data.frame(prediction = as.character(prediction), actual = as.character(actual))
      myRes <- rbind(myRes, ok)
    }

    prediction <- myRes$prediction
    actual <- myRes$actual
    output <- caret::confusionMatrix(myRes$prediction, myRes$actual)
    rm(myRes)
    output <- output$table
    output <- as.numeric(output)
    results[[i]] <- output
  }
  return(results)
  stopCluster(cl)
}


mean_and_sd <- function(x) {
  c(round(mean(x, na.rm = TRUE), digits = 2),
    round(sd(x, na.rm = TRUE), digits = 2))
```

```
}

RF_10_cross_validation <- function(x) {
  ok <- RF_10_CV(x)
  results <- data.frame(ok)
  rm(ok)
  data <- data.frame(results)
  rm(results)
  m = ncol(data)
  ACC  <- matrix(nrow = m, ncol = 1)
  SENS  <- matrix(nrow = m, ncol = 1)
  SPEC  <-matrix(nrow = m, ncol = 1)
  MCC <- matrix(nrow = m, ncol = 1)

  for(i in 1:m){
    ACC[i,1]  = (data[1,i]+data[4,i])/(data[1,i]+data[2,i]+data[3,i]+data[4,i])*100
    SENS[i,1]  =  (data[4,i])/(data[3,i]+data[4,i])*100
    SPEC[i,1]  = (data[1,i]/(data[1,i]+data[2,i]))*100
    MCC1      = (data[1,i]*data[4,i]) - (data[2,i]*data[3,i])
    MCC2      =  (data[4,i]+data[2,i])*(data[4,i]+data[3,i])
    MCC3      =  (data[1,i]+data[2,i])*(data[1,i]+data[3,i])
    MCC4  =  sqrt(MCC2)*sqrt(MCC3)


    MCC[i,1]  = MCC1/MCC4
  }
  results_ACC <- mean_and_sd(ACC)
  results_SENS <- mean_and_sd(SENS)
  results_SPEC <- mean_and_sd(SPEC)
  results_MCC <- mean_and_sd(MCC)
  rm(ACC)
  rm(SENS)
  rm(SPEC)
  rm(MCC)
  results_all <- (data.frame(c(results_ACC, results_SENS, results_SPEC, results_MCC)))
  rownames(results_all) <- c("ACC_Mean", "ACC_SD", "Sens_Mean", "Sens_SD", "Spec_Mean", "Spec_SD",
                             "MCC_Mean", "MCC_SD")
  names(results_all) <- c("10_Fold_Cross_Validation")
  return(results_all)
}
```

Performance results (Training, Cross Validation, Testing)

```
training <- RF_train_classification(real_data)
training
```

```
##           Training Performance
## ACC_Mean                 95.15
## ACC_SD                    1.34
## Sens_Mean                93.54
## Sens_SD                   2.22
## Spec_Mean                96.74
```

```
## Spec_SD                   2.18
## MCC_Mean                  0.90
## MCC_SD                    0.03
```

```
cv <- RF_10_cross_validation(real_data)
cv
```

```
##           10_Fold_Cross_Validation
## ACC_Mean                     87.22
## ACC_SD                        2.87
## Sens_Mean                    83.32
## Sens_SD                       4.28
## Spec_Mean                    91.06
## Spec_SD                       3.64
## MCC_Mean                      0.75
## MCC_SD                        0.06
```

```
testing <- RF_test_classification(real_data)
testing
```

```
##           Testing Performance
## ACC_Mean                88.53
## ACC_SD                   5.60
## Sens_Mean               83.74
## Sens_SD                  8.97
## Spec_Mean               93.45
## Spec_SD                  6.44
## MCC_Mean                 0.78
## MCC_SD                   0.11
```

Function for feature importance plot

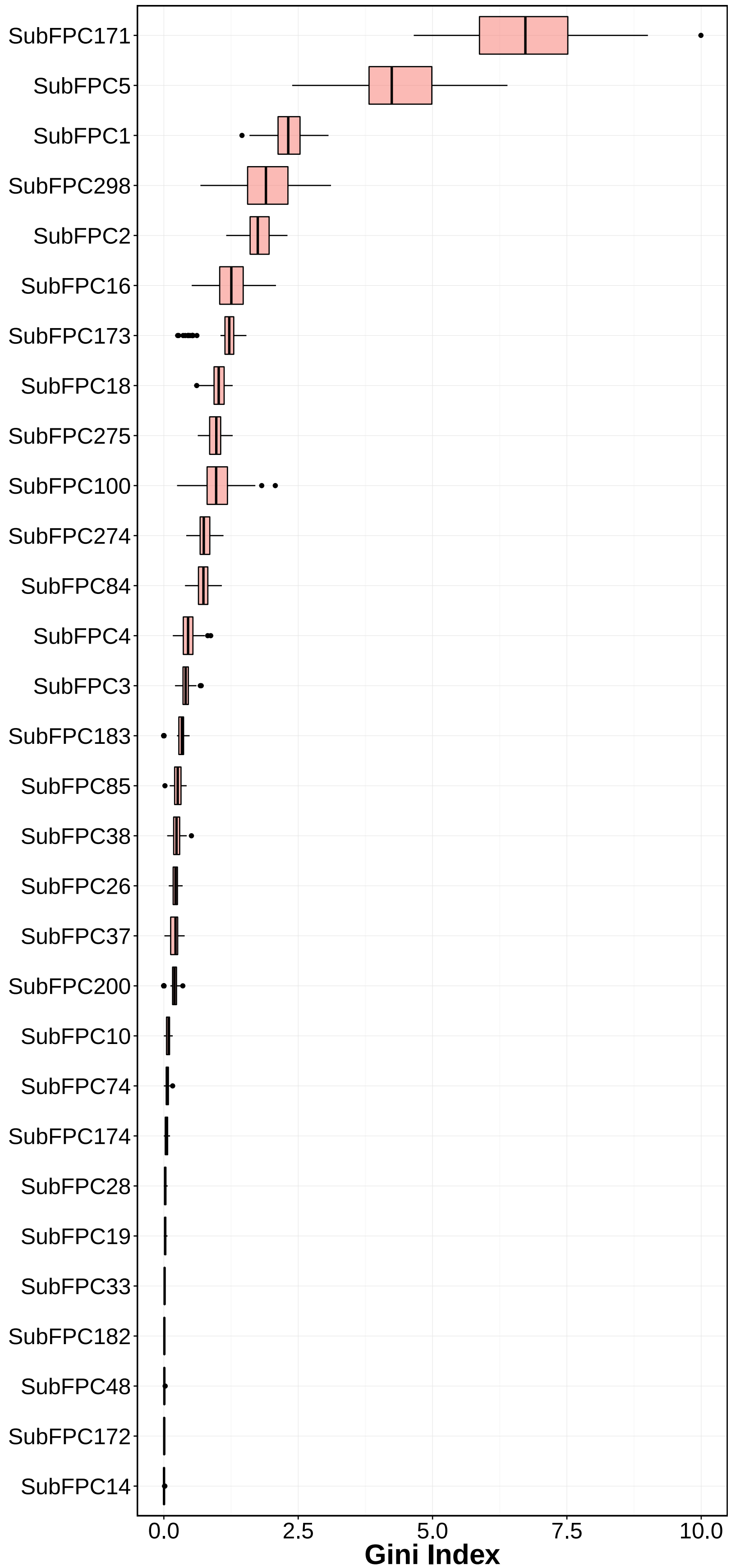
```
randomForest_feature_importance <- function(x) {
  library(doSNOW)
  library(foreach)
  library(parallel)
  cl <- makeCluster(8)
  registerDoSNOW(cl)

  results <- list(100)
  results <- foreach (i = 1:100) %dopar% {
    data <- na.omit(x)
    set.seed(i)
    trainIndex <- sample(nrow(data), size = as.integer(nrow(data) * 0.8),
                         replace = FALSE)
    Train <- data[trainIndex, ]
    Test <- data[-trainIndex, ]
    set.seed(i)
    model <- ranger::ranger(Activity~., data = Train, importance = 'impurity',
                            write.forest = TRUE, save.memory = TRUE)
    rm(Train)
    rm(Test)
    rm(trainIndex)
    importance <- model$variable.importance
```

```
    results[[i]] <- importance
  }
  return(results)
  stopCluster(cl)
}
```

Making a plot of feature importance

```
results_feature_importance_RF <- randomForest_feature_importance(real_data)
data1 <- data.frame(results_feature_importance_RF)
data1 <- cbind(features = rownames(data1), data1)
library(reshape2)
data_melt <- melt(data1, id.vars = "features")
data_melt$features <- factor(data_melt$features)
library(ggplot2)
set.seed(10)
plot_feature <- ggplot(data_melt, aes(x = reorder(features, value, FUN = median), y = value)) +
  geom_boxplot(fill = "#F8766D", colour = "black", alpha = 0.5) +
  theme_bw() + xlab("") + ylab("Gini Index") + coord_flip() + theme(
    axis.text.y = element_text(size = 20, colour = "black"),
    axis.text.x = element_text(size = 20, colour = "black"),
    plot.margin = grid::unit(c(1, 1, 1, 1), "cm"),
    panel.border = element_rect(linetype = "solid", colour = "black", fill = NA, size = 1),
    axis.title = element_text(size = 25, face = "bold", colour = "black")
  )
plot_feature
```


SubFPC171
SubFPC5
SubFPC1
SubFPC298
SubFPC2
SubFPC16
SubFPC173
SubFPC18
SubFPC275
SubFPC100
SubFPC274
SubFPC84
SubFPC4
SubFPC3
SubFPC183
SubFPC85
SubFPC38
SubFPC26
SubFPC37
SubFPC200
SubFPC10
SubFPC74
SubFPC174
SubFPC28
SubFPC19
SubFPC33
SubFPC182
SubFPC48
SubFPC172
SubFPC14
0.0
2.5
5.0
7.5
10.0
Gini Index